Win CE Integrators Guide

Revision 1.14, 30d Mar 2012
www.touch-base.com\documentation\embedded

Overview | Delivery | Components | Requirements | Using the components | Testing | Calibration
Driver settings | Stylus Notes | Diagnostics | Port issues | Known issues | Limitations | Contact

Overview

These notes refer to the UPDD CE driver 4.1.10 first released April 2011. *For customers using the earlier 4.0.6 release please refer to the instructions here*.

Windows CE or Windows Embedded Compact are embedded operating systems. For OEMs requiring a touch screen, or other pointer interface on Windows CE devices, the Touch-Base Universal Pointer Device Driver suite of software includes a 5.x, 6.x and 7.x CE driver.

With this release UPDD supports an interface with the standard Windows CE GWES (Graphics, Windowing, and Events Subsystem) touch interface. This allows for calibration via the built in CE control panel Stylus option and provides an interface to the gesture interface in Windows CE 6.0 and later. The old 'mouse' interface is retained for testing purposes and can also be used in situations whereby the GWES touch interface is not part of the CE image.

This UPDD version is built on the same software base as all other UPDD variants so that most UPDD functions available on other platforms are now supported in Windows CE (except for minor differences to accommodate variance in the Windows CE implementation). Known exceptions are listed in the limitation section below.

The UPDD Application Program Interface is supported on Windows CE allowing 3rd party utilities to be developed.

The main differences between the new and previous driver are:

- based on the new 4.1.10 driver architecture
- has been implemented as a native CE touch device driver interfacing with the standard Windows CE GWES touch interface
- utilises CE calibration procedure
- supports the native gesture interface introduced in Win CE 6
- is shipped with the new command line utility
- uses the standard UPDD settings file

History

Date	Version	CE release	Description
April 11	4.1.10	CE5 and 6	Initial X86 release for USB controllers
May 11	4.1.10	CE7	Initial X86 release for USB controllers
June 11	4.1.10	ALL	Builds for other processors available for USB controllers
Oct 11	4.1.10	ALL	EEprom support utilizes new eeprom framework for storing UPDD calibration data. Serial devices now supported. FIX - USB driver now un/reloads correctly.
Feb 12	4.1.10	ALL	USB hot plug support added

Hardware Interfaces

Serial – Available Oct 11 build onwards.

USB has been tested on X86 and ARM processors. If there are USB interface issues on other processors we may need to be supplied a target system to investigate further and modify the UPDD USB interface where applicable.

Other interfaces such as PS/2, ISA etc could be added if required.

See important Port Interface issues below.

Processor support

The driver has been tested in house with X86 and ARM CE. We can build drivers for other processors, as supported by the Microsoft CE Platform Builder, on request.

At the time of writing, the processors supported by UPDD in the various versions of CE are as follows:

Win CE version	Processor
5	X86, armv4i, mipsii, mipsii_fp, mipsiv, mipsiv_fp, (for SH4 contact Touch-Base)
6	X86, armv4i, mipsii, mipsii_fp, mipsiv, mipsiv_fp, SH4 ARM note. Based on this article which states "The ARMV6 and V7 architectures have enhanced cache designs which can improve the performance of the Windows Embedded CE 6.0 kernel. The ARMV6 processor is architecturally similar to the ARMV4I. The ARMV6 can run an OS image targeted for the ARMV4I processor" there is a possibility that the armv4i driver will run on ARMV6 architecture (ARM11 processors) – see also http://www.arm.com/community/software-enablement/microsoft/windows-embedded-ce.php, Architecture Support tab.
7	X86, armv5, armv6, armv7, mipsii, mipsii_fp

More information on the CE product range is available here:

http://www.microsoft.com/windowsembedded/en-us/develop/windows-embedded-products-for-developers.aspx

Any system/processor can be supported as long as a Board Support Package is available for the target processor. Read http://www.microsoft.com/windowsembedded/en-us/downloads/board-support-packages-for-windows-embedded.aspx

for more information on BSP's

Target hardware may have to be supplied for testing if any problems are experienced with the driver.

Important note: Although we have built the CE components for the processors listed above we can not test the driver on all supported processors due to lack of target hardware. In theory the drivers should work on the target processor as long as the component build process utilises the relevant BSP.

Software delivery

In this environment, UPDD is supplied as a number of separate components. Software sent via email will be held in the file ZIP file TBUPDDCE.ZIP.. Touch-Base utilises virus detection software on all of our systems but recipients of the software should pass the files through their own virus checking software before proceeding with installation.

Normally a software package will be supplied with just one touch controller supported in which case the settings file can be used as supplied, unless using a serial controller and the com port needs to be changed via the autoinstalldevice setting. If the software package contains more than one controller definition this setting is also used to indicate the serial controller definition to use.

Components

The UPDD CE package consists of the following components:

Component	Description
Logo.png	Suppliers logo file – not used in this release but may be in future if a CE version of UPDD Console is generated
tbupdd.bib	UPDD Configuration file entries
tbupdd.dat	UPDD System file entries
tbupdd.reg	UPDD Registry file entries
tbupdd.ini	Driver and touch device settings file.
tbapi.h	Header file (Application Programming Interface – See **)
Processor sub folder	UPDD target processor related files: X86, ARMV4I, MIPS etc
upddce.dll	Native CE touch driver.
tbupddceusb.dll	USB interface. Not required for serial interface.
tbcalib.dll / tbcalib.exe	Calibration program. Driver use only - used for the purpose of managing EEPROM calibration. Oct 11 onwards renamed to .exe. Cannot be used for normal calibration! March 2012 – Can be used to calibrate in system missing CE stylus calibration
tbutils.exe	Command line interface.
ace.dll	Inter process communication library for the target processor
tbapi.dll	API function calls (Application Programming Interface – See **)

** Under CE the UPDD Application Programming Interface is implemented as a Dynamic Link Library TBAPI.DLL and function declaration file TBAPI.H. These are supplied with the driver but documented here

Requirements

The basic system requirements for utilising UPDD are as follows:

- Supported CE version: 5, 6 or 7.
- Supported processor version or one where a BSP program can be supplied. Alternatively x86 based images can be run on a standard PC using the LOADCEPC utility.
- Supported touch controller
- The driver's touch interface into CE is via GWES **or** standard mouse port.
  To utilize the native CE touch interface (GWES) and calibration utility the target CE system must contain the touch screen CE components. If these components are not part of the image the 'mouse' interface is used.
- The target system image must have been created with the appropriate platform builder with any relevant system packs. We have had a customer report that he was showing 'crazy' errors in the CE log file and identified the problem as being a CE update applied to his Visual Studio 2005. He rolled back the updates and the error disappeared!

  For internal testing we build images with VS2005 with the relevant system packs applied but with NO subsequent CE updates applied.

Specific embedded or post install requirements are specified in the relevant section below.

Using the components

The components can either be installed on an existing CE system or embedded into a CE image as detailed below.

Installing on existing image

Installing into an already created image can be performed for two reasons, either to test / demo the driver prior to embedding or as a permanent installation solution whereby there is not an option to embed the driver into the desired image.

If installing the driver for test or demo purposes then the driver can be installed, tested/demoed and then discarded. However, if the installation is being used to install the driver as a fully working solution this can only be achieved on systems that have persistent storage for the file system and the registry such that the changes to the system are retained over a reboot and available during driver load.

and the registry such that the changes to the system are retained over a reboot and available during driver load.

These notes are based on installing on a CE5 image that did not contain the touch stylus components built into the image and therefore the driver was configured to use the basic mouse interface to handle cursor movements and click.

Whether a post installation is possible for a given system depends completely on the architecture of that specific system and in some cases may simply not be possible. However these notes can be used as a starting point to experiment.

As a minimum the system will need:

1) Persistent disk storage, such as a flash drive or similar device.
2) A mechanism to save updates to the registry again typically on a flash device.
   A customer reported using an image whereby the manufacturer of the board had embedded a utility that periodically wrote any registry changes to the flash drive.
3) A mechanism to load files onto the target system.

You will also need a tool to edit the registry. One such editor is described here

http://geekswithblogs.net/BruceEitman/archive/2009/07/27/window-ce-simple-little-registry-editor.aspx

Without these aspects the software will probably not work in a post installation scenario. There may well be other dependencies which prevent this working so we cannot guarantee that post installation will be successful and this will need to be assessed on a case by case basis.

The steps to follow are:

1) Identify the system path by examining registry entry HKEY_LOCAL_MACHINE\Loader\SystemPath or HKEY_LOCAL_MACHINE\Launch\SystemPath for the system path entry. This lists the folders searched by the system for .dll or .exe files. Identify if one of these folders is on your persistent storage device which can be used for the driver modules and settings file or create a new folder and add its location to this registry entry. If you need to add additional folder names to the system path you will need a registry editor, such as ceregeditor or SLRE.

   It is our understanding that most CE systems will have a SystemPath. However, if this is not the case and you have to create one ensure the key is created as type REG_MULTI_SZ.

2) The driver's touch interface into CE is via GWES **or** standard mouse port. The GWES interface will only be available if the image was built for touch and includes the Touch Screen (Stylus) components. If the touch component is not built into the image (the stylus calibration option in the control panel will be missing) then enable the mouse port interface.

3) Extract the supplied updd files and copy to the target system by whatever means are appropriate, copy to the folder described in Step 1 above. The files *.bib, *.dat, *.h *.reg are not required on the target.

   If copying over a previously installed UPDD driver then the system will report certain UPDD files are in use and cannot be overwritten. Renaming the file will allow another copy to be placed on the system – e.g. 'ren upddce.dll deleteme' – which can then be deleted following the next reboot and once the updates have been tested.

   When extracting the driver software some files will be placed in a sub folder that represents the processor type, i.e. X86. One customer implied that they had issues when using the sub folder structure and placed all files in the same folder. We did not experience any issues when using the sub folder structure.

4) Take the file tbupdd.reg and add all the registry settings from it to the registry on the device, using a suitable registry editor.

5) Create a further registry setting

   HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd\settings=<path to ini>. This identifies the full path to tbupdd.ini

Embedding

It is assumed that the developer is using the appropriate Visual Studio, Platform Builder and service packs to create the relevant Win CE image for the target hardware.

The software is supplied in a form to make it easy for users with little experience with Windows CE to get up and running as quickly as possible. CE experts with knowledge of the BIB structure are free to amend the configuration files as required, so long as the embedded files are located in accessible locations on the target and with the correct attributes.

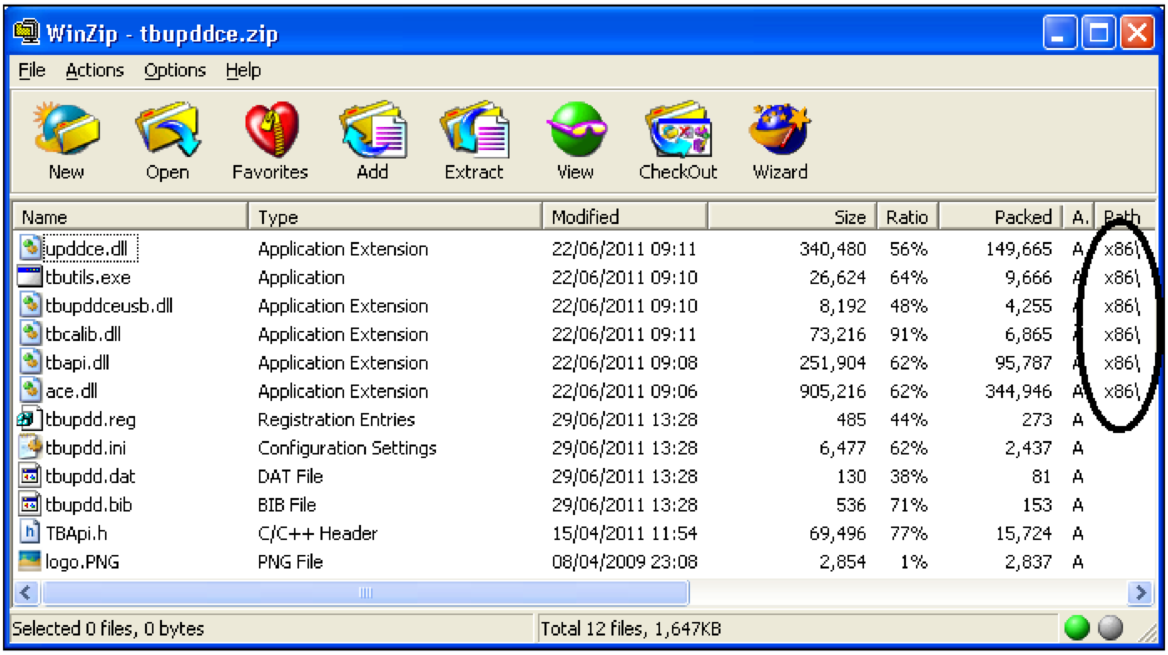
For users unfamiliar with Windows CE we suggest the following guides to creating a Windows CE image.

http://msdn.microsoft.com/en-US/library/ee483161(v=WinEmbedded.60).aspx (CE 6 / 7)

http://msdn.microsoft.com/en-us/library/aa446910.aspx (CE5)

To incorporate the driver into your image follow these instructions:

1. Expand the tbupddce.zip file. Some of the files are target specific, these target specific files are shown by the black highlight in the example below (in this case for the x86 target):



Name	Type	Modified	Size	Ratio	Packed	A.	Path
updd ce.dll	Application Extension	22/06/2011 09:11	340,480	56%	149,665	A	x86\
tbutils.exe	Application	22/06/2011 09:10	26,624	64%	9,666	A	x86\
tbupddceusb.dll	Application Extension	22/06/2011 09:10	8,192	48%	4,255	A	x86\
tbcalib.dll	Application Extension	22/06/2011 09:11	73,216	91%	6,865	A	x86\
tbapi.dll	Application Extension	22/06/2011 09:08	251,904	62%	95,787	A	x86\
ace.dll	Application Extension	22/06/2011 09:06	905,216	62%	344,946	A	x86\
tbupdd.reg	Registration Entries	29/06/2011 13:28	485	44%	273	A	
tbupdd.ini	Configuration Settings	29/06/2011 13:28	6,477	62%	2,437	A	
tbupdd.dat	DAT File	29/06/2011 13:28	130	38%	81	A	
tbupdd.bib	BIB File	29/06/2011 13:28	536	71%	153	A	
TBApi.h	C/C++ Header	15/04/2011 11:54	69,496	77%	15,724	A	
logo.PNG	PNG File	08/04/2009 23:08	2,854	1%	2,837	A	

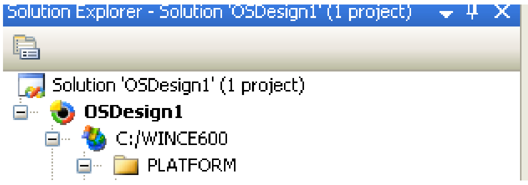
Selected 0 files, 0 bytes

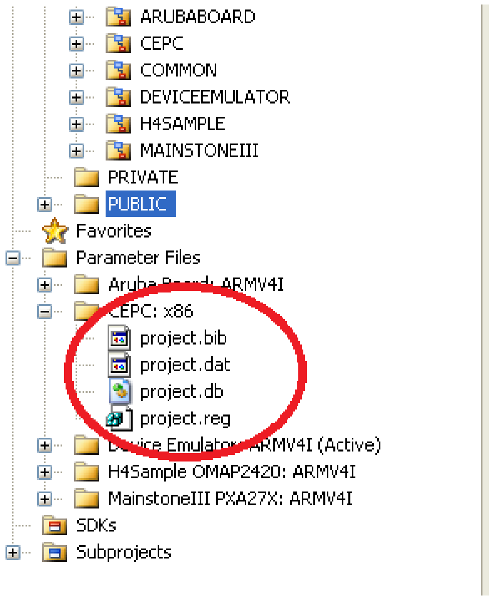
Total 12 files, 1,647KB

2. Copy the files for your chosen target (processor) to the root folder of your build system (typically c:\wince500, c:\wince600 or c:\wince700).

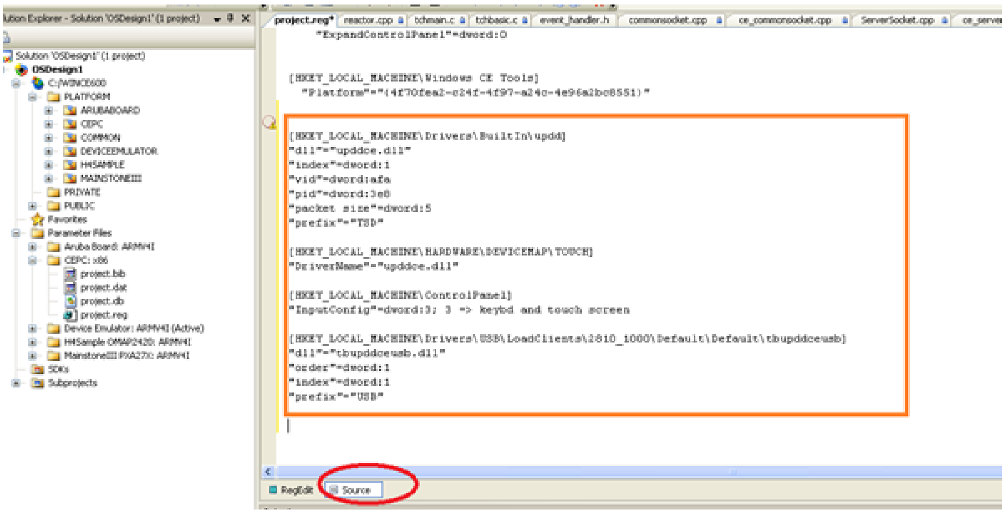
   Also copy the file tbupdd.ini to the same location

3. Locate the files project.bib, project.reg and project.dat in your platform builder (visual studio in CE 6.0 and later) project tree. Open the files tbupdd.bib, tbupdd.reg and tbupdd.dat and copy / paste the contents at the bottom of the corresponding project files


Solution Explorer - Solution 'OSDesign1' (1 project)
Solution 'OSDesign1' (1 project)
OSDesign1
C:/WINCE600
PLATFORM



The following example shows the updd settings added to the project.reg file. Note in this case the source tab is selected.

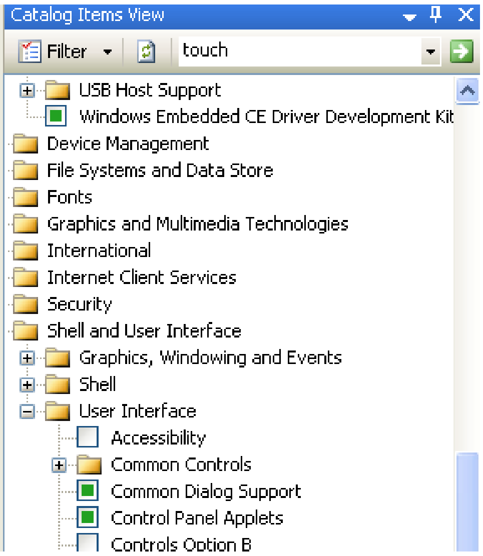


```
  "ExpandControlPanel"=dword:0


[HKEY_LOCAL_MACHINE\Windows CE Tools]
  "Platform"="{4f70fea2-c24f-4f97-a24c-4e96a2bc8551}"


[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd]
"dll"="updddce.dll"
"index"=dword:1
"vid"=dword:afa
"pid"=dword:3e8
"packet size"=dword:5
"prefix"="TSD"

[HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\TOUCH]
"DriverName"="updddce.dll"

[HKEY_LOCAL_MACHINE\ControlPanel]
"InputConfig"=dword:3; 3 => keybd and touch screen

[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\2810_1000\Default\Default\tbupddceusb]
"dll"="tbupddceusb.dll"
"order"=dword:1
"index"=dword:1
"prefix"="USB"
```

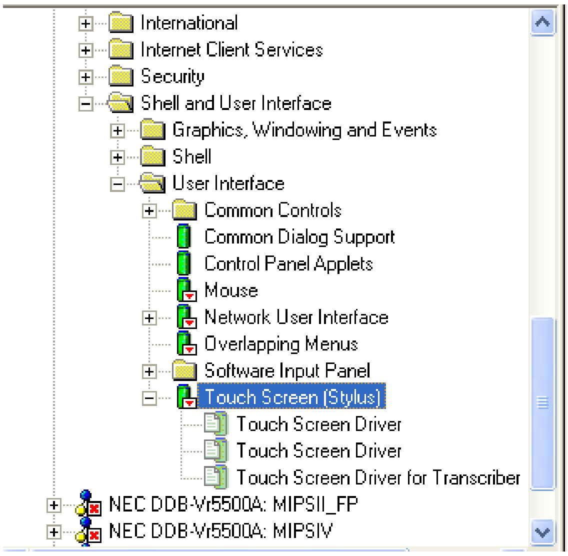
4. In order to utilize the native touch interface and calibrate the screen via the stylus calibration option in the control panel the Touch Screen (Stylus) option is selected in the design view / project catalog. If this component cannot be embedded it is important to enable the mouse interface.

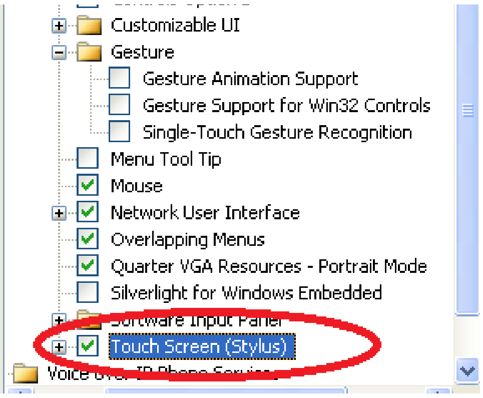
   This option enables stylus functionality within the CE image. A certain amount of input functionality, such as the Software-based Input Panel, Handwriting Recognizer Engine (HWX), and Transcriber Handwriting Recognizer Application Catalog items, require the Touch Screen (Stylus) Catalog item. UPDD utilizes the control panel Stylus option to perform calibration.

CE6 touch stylus enable



CE5 touch stylus enable (**see important note below**)




Customizable UI
Gesture
Gesture Animation Support
Gesture Support for Win32 Controls
Single-Touch Gesture Recognition
Menu Tool Tip
Mouse
Network User Interface
Overlapping Menus
Quarter VGA Resources - Portrait Mode
Silverlight for Windows Embedded
Software Input Panel
Touch Screen (Stylus)
Voice over IP Phone Services

Important note: In our tests this turned out not to work correctly. We circumvented this issue by making the following change, which might be useful if a similar problem is encountered.

1) Open the file C:\WINCE500\PUBLIC\CEBASE\OAK\MISC\wceshellfe.bat

2) Locate the line

if "%__SYSGEN_TOUCH_CURSOR%"=="1" set CPLMAIN_COMPONENTS=%CPLMAIN_COMPONENTS% stylus

3) Edit this line to read

set CPLMAIN_COMPONENTS=%CPLMAIN_COMPONENTS% stylus

Having followed the above instructions you should now be able to build and test your CE image.

Additional embedding notes:

1. Registry entries
   - If using a USB device then one of the entries in the .reg settings refers to the device and will contain the USB Vendor and Product ID values. The VID and PID values MUST match the VID and PID of the controller in use, e.g. Controller has hex VID = **1234** and PID = **11** values. Hex 1234 = decimal **4660** and Hex **11** = decimal **17**. Based on this example the settings would be as follows:

     project.reg
     [HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients\4660_17\Default\Default\Tbupddceusb]

     tbupdd.ini
     [updd\parameters\1]
     Product id=0x00000011
     Vendor id=0x00001234

     Important note: The files supplied should have the correct vendor and product ids as they will have been generated for a touch device as requested. If the values in the registry do not match the values of the usb controller then request a driver that does match. If you manually change these values to match then the driver will connect to the device. However, the touch data generated is unlikely to match that as expected so it is likely the touch will not work as expected.
   - Ensure the InputConfig entry is updated to indicate a touch screen is in use. For more information on this setting see http://msdn.microsoft.com/en-us/library/ee482243.aspx
2. If, for any reason, the GWES component cannot be utilised in the image then it is important to enable the 'mouse' interface.

3. Modify any settings in the UPDD settings file as required. See Driver Setting section below for more information.

4. If system does not have persistent registry or file system determine appropriate calibration strategy as described in the calibrate section below.

5. Make any required software changes to the system components. See "Port interface issues" below.

Testing

One feature of the CE touch interface is that there is no mouse cursor shown. This is by design as in a touch environment the visual feedback is at the point of touch. The easiest and quickest test is to touch on the CE desktop – when dragging on this screen a "wire frame" marker is seen. This is particularly useful when working with an uncalibrated or unattached touch screen.

Calibration

The touch screen needs to be calibrated with the desktop such that the point of touch generates a touch at the correct position on the desktop. The calibration procedure generates touch co-ordinate data that is associated with known positions on the desktop. Using this data the driver can calculate the correct desktop position from incoming touch co-ordinate data. This calibration data can be both generated and utilised in a number of ways as discussed below. The method you select will be dependant on your system's configuration.

Options

Based on your requirements you may choose any combination of the following calibration options to cater for calibration within your system.

Option	**Description**	
Manual	A calibration procedure is invoked to show on-screen calibration points that are touched to collect calibration data	
	CE Stylus	Using the build in stylus calibration function. A user invokes this function from the control panel, stylus, calibration option. This method offers tight integration with Windows CE. This is only available on systems built with touch support enabled so is not suitable for "post installation" unless the image already has touch support enabled. It is currently also restricted to a 4 point calibration with a 10% margin.
	UPDD Calibration	Using the UPDD calibration program. Tbcalib.exe provides most calibration features available to the desktop based calibration software (apart from some custom graphic features and eeprom support for some controllers.This module is not integrated with CE (e.g. in the control panel) by default although of course an integrator is free to adopt an appropriate integration strategy to suit the target device.

Pre-defined	The supplied UPDD settings file does not contain any calibration data so devices are uncalibrated when first used. However, it is possible to place good calibration data in the settings file such that the device is calibrated following image creation or install.	
EEPROM storage	UPDD data	Where supported, caters for UPDD calibration data to be stored in the controller's eeprom and not dependant on any calibration data being available on the CE system.
	Hardware based	Invokes controller based calibration procedure – not supported in new driver but will be available in a future release.

Manual Calibration.

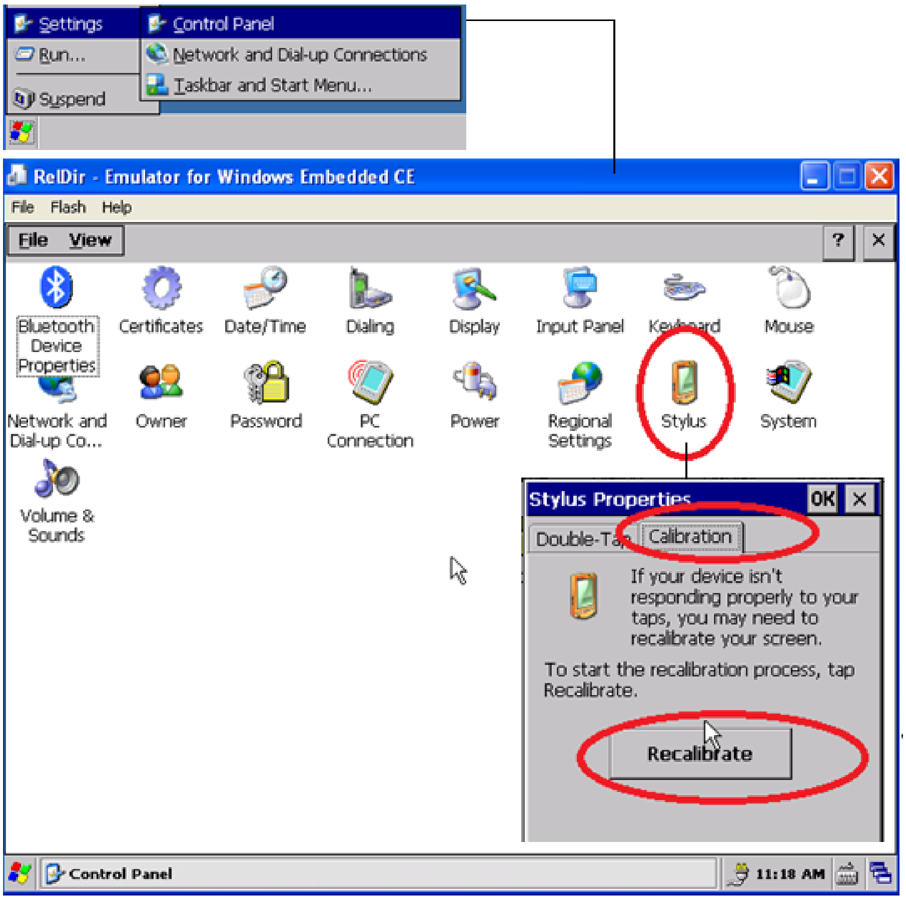
The native stylus or UPDD calibration functions are invoked to perform manual calibration. An implementation using manual calibration needs to decide on a strategy for initiating the calibration procedure. E.g. executing the calibration program at start-up or placing an icon on the desktop.

These and other options are implemented via the platform configuration.

One option we are considering is that the driver will automatically invoke the calibration procedure at startup based on a system setting. This would be used in environments where calibration was performed every time the system started. Please contact us if this is required.

Native CE stylus

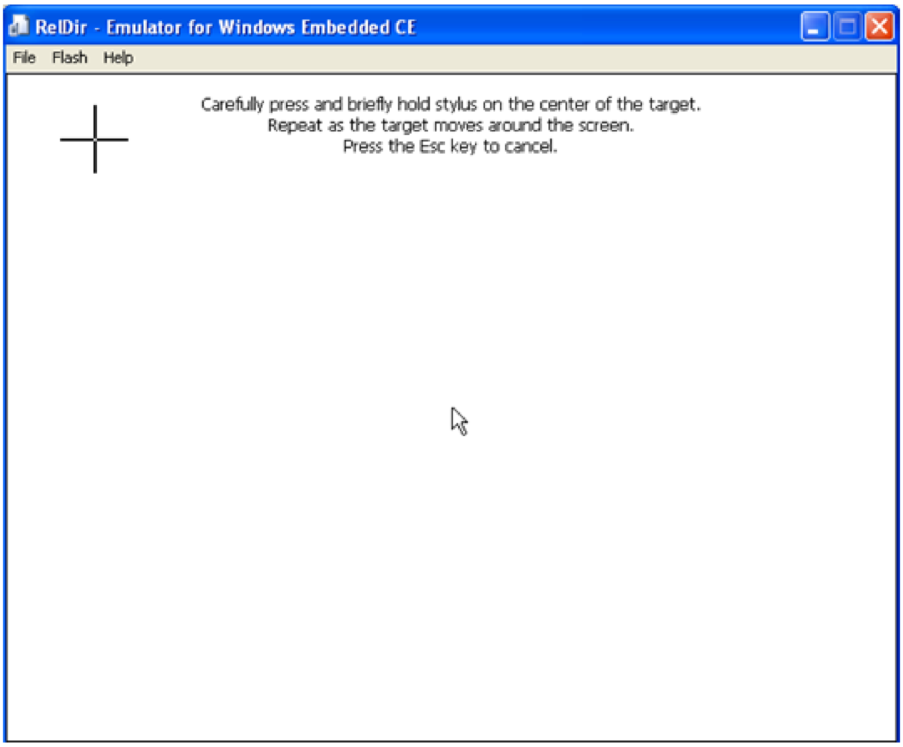
Once the image is running you can calibrate the touch screen by using the Stylus option in the control panel. This is illustrated below. Note that the exact screens vary according to the system configuration.



If your device isn't responding properly to your taps, you may need to recalibrate your screen.

To start the recalibration process, tap Recalibrate.

This will invoke the calibration procedure:



Carefully press and briefly hold stylus on the center of the target.
Repeat as the target moves around the screen.
Press the Esc key to cancel.

The native calibration procedure will store calibration co-ordinates in the registry at HKEY_LOCAL_MACHINE\Hardware\DeviceMap\Touch\CalibrationData. Given that the UPDD driver is a GWES conformant driver the native calibration calls the SetCalibrationPoints function within our driver. The calibration data is converted to the format used by UPDD and stored in the UPDD settings file. If eeprom storage is enabled and configured for UPDD data storage then the calibration data is also written to eeprom.

We understand that registry setting HKLM\HARDWARE\DEVICEMAP\TOUCH\MaxCalError deals with how picky the touch calibration application will be when you touch the calibration crosses.

UPDD Calibration function

Once the image is running you can calibrate the touch screen by using the TBcalib program.

The calibration data is stored in the UPDD settings file.

If eeprom storage is enabled and configured for UPDD data storage then the calibration data is also written to eeprom.

If eeprom storage is enabled and configured for hardware eeprom the associated hardware calibration procedure is performed such that the controller scales the co-ordinate output accordingly.

Pre-define calibration data

In cases where neither persistent storage nor EEPROM based calibration are available the calibrated data can be embedded in the target image to pre-calibrate the system. This approach is only suitable for systems where the calibration data does not change significantly over time or with use.

To pre-calibrate the system:

1) Calibrate normally

2) From a command prompt (cmd.exe) type "tbutils dump4tba". – for more information see Command Line Interface document

3) Open the file tbcalib.tba created by this process and copy the string contained in it (if no file is created ensure you are running the progr am in a writable folder and try again, e.g. /application/data/updd).

4) Paste the string into the calibration styles section for the controller replacing the default info shown in red below. This is in the copy of tb updd.ini in your build environment:

```
[updd\parameters\controller\ts001]
....
calibration styles=Normal,n,n,n,n,n,n,n,n,n
```

5) Adjust the following settings to cater for any calibration inversion required:

```
[updd\parameters\controller\ts001]
invertx=0x00000000
inverty=0x00000000
swapxy=0x00000000
```

6) Rebuild the target image.

If correct calibration data is stored in the settings file then the default settings will offer a calibrated system.

If correct calibration data is stored in the settings file then the default settings will offer a calibrated system.

We suggest that precalibration in the registry be avoided as these values interact with other items (e.g. will override file based persistence) and the format of this data could possibly change with future UPDD releases.

EEPROM storage

Some controllers support the saving of calibration data in persistent memory (eeprom) on the controller itself. Subject to the controller supporting this feature and UPDD implementing EEPROM support for the specific device then calibration data can be saved on the device.

If the UPDD setting file has an entry 'eeprom protocol = 'protocol id' then this indicates that updd supports eeprom storage for the touch controller but it does not necessarily indicate if this is supported in CE due to differences in the CE platform. See EEprom documentation for more information about supported controllers under CE.

For supported controllers simply set the eeprom calibration option in tbupdd.ini to 1

```
[updd\parameters\controller\ts001]
```

```
eeprom calibration=0x1
```

eeprom protocol='protocol id' – This is the protocol id used by the driver to support eeprom in the controller.

For the eeprom data to be read from the device automatically at startup the following registry setting must also be set;   eepromreadatstart=1; e.g.:-

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd]
```

```
" eepromreadatstart "= dword:1
```

Important Note, reading the eeprom data at startup loads an additional module (tbcalib.exe) into kernel memory so in some cases the memory mapping layout of the image may need to change. We most often see this in debug images. If enabling this option leads to a crash in tbcalib at startup this is very likely the cause.

For hardware based eeprom calibration (where supported) the controller's coordinate touch data will be scaled accordingly.

Calibration persistence.

A common issue with CE devices is saving calibration data when the device is reset or restarted. There are 3 choices available. Which if any of these are possible depend on the touch controller hardware and the setup of the CE device.

1) Registry persistence. If the device registry is on persistent storage then updd will use the registry to save calibration data. This is currently only supported when stylus based calibration is used (not tbcalib.exe). The data is saved in HKLM\HARDWARE\DEVICEMAP\TOUCH\CalibrationData. The format of this data is defined internally and subject to change.
2) File system persistence. If the control file tbupdd.ini is on persistent storage then updd will use this file to save calibration data. This is supported for Stylus and tbcalib based calibration. It may be necessary to locate the tbupdd.ini file at an alternate location in which case this setting can be used. Note, if both registry and file based persistence are available updd will utilise the registry based data although the data will be written to both locations.
3) EEPROM storage. Subject to the controller supporting this feature and UPDD implementing support for the specific device calibration data can be saved on the device.

Unless the target system has persistent storage for the registry, or the file system holding tbupdd.ini, or eeprom storage is used then calibration data will not be saved across a reboot.

If persistent storage is available for the tbupdd.ini or registry file then no further action is required. In the case that a persistent registry is available the software saves the calibration data to the registry and this data is written to tbupdd.ini at startup.

However, if calibration data is not held in permanent or non volatile memory then additional calibration options, such as pre-calibration, should be considered.

Calibration beeps

Calibration beep option to be investigated and not currently available. Please contact Touch-Base should this be required.

Calling calibration from an application

This section to be confirmed as is untested and speculation only. It has been copied from an article on the internet…

Sometimes it may be necessary to cause a touch calibration to occur from an application, perhaps because of drift or because the initial values as set during the production process weren't quite right.

This is fairly straightforward to achieve as the relevant method is exposed by `coredll.dll`

All that is needed is to declare a platform invoke (P/Invoke) to the `coredll.dll` method, which can then be called from within application code:

A C# compact framework example would be:

```
using System;
using System.Runtime.InteropServices;

namespace DynamicDevices.Utilities
{
    ///
    /// Expose 'coredll' native methods
    ///
    public class NativeMethods
    {
        [DllImport("coredll.dll")]
        public static extern void TouchCalibrate();
    }
}
```

Rotate

The current version of the driver supports operation in a fixed rotated mode. All that is required is to calibrate in the desired rotation.

This article http://msdn.microsoft.com/en-us/library/ms914404.aspx describes how to set Portrait mode as default if required

Dynamic rotation is not yet supported. Given that GWES is used and is responsible for rotation it is possible that this will also just work, but this is subject to confirmation.

Driver settings and general notes

The driver's settings are located in the UPDD settings file and will usually be set to the default settings for the controller in use. These settings are defined as part of generating the CE driver for a given touch screen device.

Given that all settings for an embedded system are defined in the CE Platform Builder and the image then embedded on the target image it has been decided that there is no need for a UPDD CE GUI as settings in these environments are locked down and in most cases the supplied configuration file will contain the required device and driver settings. If this is not the case then settings can be manually edited in the settings file as required. We also offer a command line interface utility that can be used to change settings on the fly.

On the CE target system there will be 2 copies of tbupdd.ini, the location of which is dictated by the contents of the tbupdd.dat file.

\windows\tbupdd.ini or \Windows\tbupdd.ini.orig (Oct 11 onwards)
\application data\updd\tbupdd.ini

This is because the BIB script creates the first entry and the DAT script then creates the second real working entry.

The working copy has writeable hence placed in \application data. Should you require this to be in a different writable location modify the tbupdd.dat file accordingly and then define the following registry entry:

HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd\settings=<path to ini> which identifies the full path to tbupdd.ini

Pre Oct 11 - We did not find a method to eliminate the \windows copy. The redundant copy can be ignored and in any case is very small.

Post Oct 11 - We determined it is not possible to delete the \windows copy so we now copy the file to \Windows\tbupdd.ini.orig and this is in turn copied to the active area. This is handled by the .dat file so no user action is required.

Touch interface via mouse port

The driver's touch interface into CE is via GWES **or** standard mouse port. The GWES interface will only be available if the image was built for touch and includes the Touch Screen (Stylus) components. If the touch component is not built into the image (the stylus calibration option in the control panel will be missing) then enable the mouse port interface as follows:

Edit the supplied settings file tbupdd.ini to add this entry:

[updd\parameters] <- after this line
cesendinput=0x00000001 <- add this entry or amend if it is already defined

This setting instructs the driver to use the standard mouse interface and might not be necessary if the system supports a GWES touch interface. If unsure try with and without this setting.

Important note: At the time of writing updd uses the GEWS interface for calibration via the stylus calibration option in the control panel. If the target does not have this enabled then only pre-calibration is currently supported. Pre calibrated data should be embedded in the settings file as described in the calibration section below, albeit the calibration may need to be conducted on a different system, say a Windows desktop, to produce the pre-calibration data that can then be embedded. A standalone calibration utility is scheduled for a future release.

Serial port device and com port selection

The setting autoinstalldevice is used to direct the installation of the device through the UPDD PNP manager. For USB devices this is handled automatically but for serial devices this setting is used to select the required serial device and specify the com port, even when only one is configured in the software package.

This setting is described below.

Example

[updd\parameters]
autoinstalldevice=1¬Microchip, AR1100, Serial¬¬COM1

Description

autoinstalldevice is comprised of several fields separated by the "¬" symbol as described below:

1) The controller id. Usually 1. This tells the PnP manager which TSNNN (default controller settings – one per supported controller in the settings file) entry to select during installation of the device. This value should not be changed unless you have a software package with multiple controllers and you wish to select a controller other than the first defined for installation. E.g. to select TS003 set this to 3.
2) The name with which the device will be installed. This can be freely edited.
3) Unused (needed for compatibility with the standard updd installer).
4) The COM port. This can be edited as required. This is the com port that will be selected when the device is installed. See important Port Interface issues below when defining serial ports.

Sound

Not currently supported in the new driver. Please contact us should this be required.

Mouse Mode

As standard the GWES touch interface does not utilize the mouse cursor therefore there is no mouse cursor movement when touching the screen. In fact, when using the touch interface the mouse cursor becomes invisible. For test purposes, during the development of the new native touch interface we retained the previous mouse interface (which utilises the SendInputAPI) and this can be enabled if mouse emulation (and therefore mouse cursor movement) is required. It should be noted that operating systems are moving away from the system cursor when utilising touch and we believe more end users should be encouraged to use touch without the traditional mouse cursor utilized for visual placement feedback. Mouse cursor is Mouse interface, other or no visual feedback is Touch.

However, should you wish to see mouse cursor movement then the following UPDD setting will enable the old mouse interface:

```
[updd\parameters]
cesendinput=0x00000001
```

It should be noted that any touch functionality built into the OS which is enabled when touch input is processed via the GWES touch interface will be lost if the touch data is passed via the mouse interface. This is likely to be more relevant to Windows Compact (CE 7) than pre CE 7 versions.

We believe that for mouse mode to work you will need to configure the Mouse Catalog in your CE image.

Priority levels

In order to provide adequate performance it is necessary for the driver to execute at a higher priority than other active processes. In particular the explorer component consumes a lot of CPU in some cases, so dragging items on the desktop can be slowed. By default all critical threads in the driver execute at priority level 249 which gives good results in most cases (which was the same as the HID mouse).

Should you wish to alter this priority for any reason you can change the following settings.

For the main driver (upddce.dll) add the Priority setting in tbupdd.ini setting in the [updd] section.

```
[updd]
....
priority=0x000000FA
```

For the USB interface component (tbupddceusb.dll) add the registry setting

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd]
"priority"=dword:fa
```

Touch packet data rate

In years gone by it had been observed on some lower end systems that if all the incoming data packets from fast touch controllers are processed it can have a detrimental effect on the performance of the system. By default the driver processes 100% of the packets. This percentage can be adjusted by the UPDD setting [updd\Parameters\1\]:SampleRate.

Changing the setting from 64hex (100 dec) to 32hex (50 dec) indicates to process 50% of incoming packets.

Right Click processing

A "right click on hold" feature in Windows CE is provided by the WinCE component **AYGSHELL**. Windows CE generates this right click automatically when the left touch is held stationary for a certain period of time. There is nothing that the driver can currently do to influence this. If this component is present in a Win CE image the right click on hold feature is enabled and you will see the progress indicator if you hold a touch steady for a period of time.

When building an image AYGSHELL may or may not be included depending on your requirements.

If installing the touch driver on an already created image you may or may not have right click processing depending on the inclusion of AYGSHELL in the image. If you do have right click processing and wish to disable or adjust the right click threshold we are currently unaware of how this is achieved but there are references on the web to a registry entry that can be used to adjust the time “This amount of time can be controlled by the OEM through a registry setting” but we’ve been unable to find it!

Further, this article http://msdn.microsoft.com/en-us/library/aa925176.aspx implies that applications can specify if right click processing is enabled within an application.

Multi-monitor support

Windows CE does not support multiple monitors. However UPDD will support a configuration where 2 or more monitor and associated touch-screens are utilized in a “shadow” configuration, i.e. where both monitors show the same image. No special UPDD configuration is required for this mode, each additional controller detected will bind to the same device definition. Calibration can take place on either monitor. It is assumed that calibration on one monitor is good for the other so the devices must be identical.

Multi-controller definitions

In some cases you may receive a package that supports multiple controller definitions. In this case you will see multiple entries listed in tbupdd.reg in sections beginning:

```
[HKEY_LOCAL_MACHINE\Drivers\USB\LoadClients...
```

This occurs when a controller case have multiple VID or PID designators.

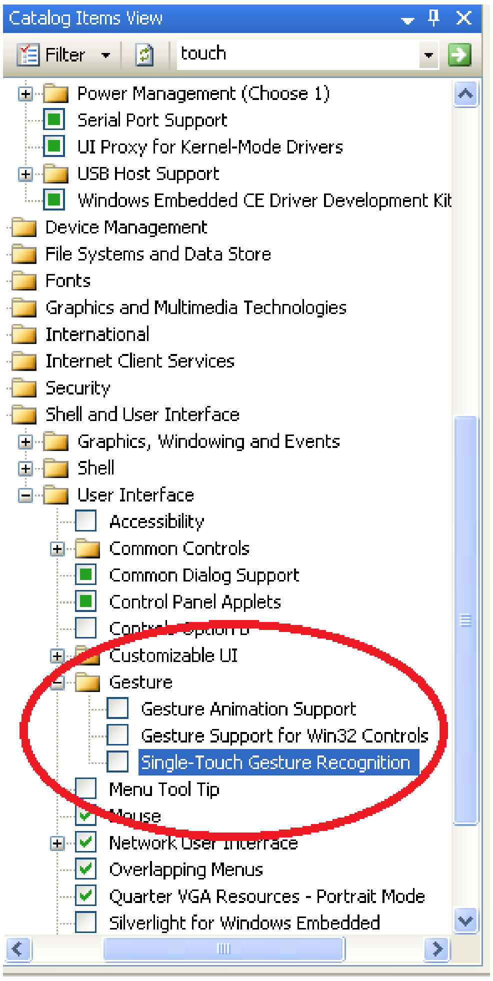
In this case the default tbupdd.reg file must be amended so that the bootstrap VID and PID entries (shown here)

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\updd]

"vid"=dword:afa
"pid"=dword:3e8
```

Match the controller in use. The default settings will match the first listed controller. NB the bootstrap entries are in HEX whereas the LoadClients entries are in decimal.

Gestures

Starting with CE version 6.0 Windows CE supports gesture functionality allowing gestures from the touch-screen to be passed to applications that support such gestures. This functionality can be enabled by selecting the items shown below in the CE catalog. This functionality is part of GWES and so cannot be used when in Mouse Mode is used. It is only useful when applications are in use that support this functionality.


Catalog Items View
Filter
touch
Power Management (Choose 1)
■ Serial Port Support
■ UI Proxy for Kernel-Mode Drivers
USB Host Support
■ Windows Embedded CE Driver Development Kit
Device Management
File Systems and Data Store
Fonts
Graphics and Multimedia Technologies
International
Internet Client Services
Security
Shell and User Interface
Graphics, Windowing and Events
Shell
User Interface
☐ Accessibility
Common Controls
■ Common Dialog Support
■ Control Panel Applets
☐ Control Option D
Customizable UI
Gesture
☐ Gesture Animation Support
☐ Gesture Support for Win32 Controls
☐ Single-Touch Gesture Recognition
☐ Menu Tool Tip
☑ Mouse
☑ Network User Interface
☑ Overlapping Menus
☑ Quarter VGA Resources - Portrait Mode
☐ Silverlight for Windows Embedded

An interesting article describing this functionality is available at http://www.ptgsystems.com/blognet/post/2009/10/20/Touch-Gestures-and-the-Physics-Engine-in-Windows-Embedded-CE-60-R3.aspx

Another comprehensive gesture article: http://blogs.msdn.com/b/marcpe/archive/2009/06/29/let-s-talk-about-touch-part1.aspx

Touch gestures in WEC7 are further described here http://msdn.microsoft.com/en-us/library/ee499124.aspx.

Toolbars

A toolbar is an area on the touch screen that acts independently from the main calibrated video area. A toolbar can simply be used to mask off areas of the calibrated video area or they can be used to trigger an event. Toolbar utilisation is described in the Toolbar document. To utilize in a CE environment the toolbars would need to be configured in a OS using the UPDD Console, Toolbar dialog and then embed the toolbar entries in the settings file into the CE settings file.

Touch Stylus Notes

Settings

We are aware that Microsoft publishes a number of settings that relate to the GWES touch stylus and we are unsure if these settings relate to all versions of CE or have in some cases been superseded by newer versions of the OS. Nevertheless we are aware that various settings are documented that may or may not affect stylus usage. Experimentation would be necessary to determine the affect they have on the stylus interface. One such reference to stylus registry settings is here.

Double Click Setting

One interesting entry is the dblTapDist setting at HKEY_CURRENT_USER\ControlPanel\Pen documented as “Default setting is 20 pixels. Indicates the size of a reference rectangle constructed around the location of a stylus tap. If a second tap occurs inside the rectangle within the allowed time, it is considered a double-tap.”

This Win CE 6 link also references dblTapTime but indicates it is only used with the Thin Client shell instead of the Standard shell. It is documented as “Indicates the time between clicks in a double-click action. The default value for a thin client is 256 milliseconds or 0xFF in hexadecimal format.”

Usage

A good overview of CE touch related issues can be viewed here.

Diagnostics

If you've embedded the software and the touch does not work there are several things you can check. Several of these require an active debugger connection, such as that provided by Platform Builder.

1) Check for known issues

Check the known issues section.

2) Manually reload the driver

From a command prompt (cmd.exe) type "tbutils reload"

This operation requires a connection to the driver process (implemented in upddce.dll). If a message indicates that this connection cannot be made then recheck that upddce.dll is in place and check all registry settings.

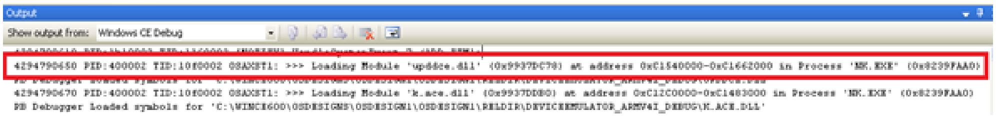
If you are using a USB controller and a message box pops up at startup asking for a driver for a specified device, then check if this message occurs without the controller plugged in. If not then this suggests an error in the registry settings, pay particular attention that the vendor id and product id match the controller in use. See the multi controller definitions section later in this document if you have more than one controller listed.

3) Check the UPDD modules loaded

Check in the debug log that both upddce.dll and tbupddceusb.dll (if using USB) are loaded and not subsequently unloaded. Typically the module load entry will read:

```
PID:00400003 TID:03770006 OSAXST1: >>> Loading Module 'tbupddceusb.dll' (0xC0845838) at address 0xEE5B0000-0xEE5B6000 in Process 'NK.EXE' (0x85302A60)
```

As see in the debug log below:



```
4294790650 PID:400002 TID:10f0002 OSAXST1: >>> Loading Module 'upddce.dll' (0x9937DC70) at address 0xC1540000-0xC1662000 in Process 'NK.EXE' (0x8239FAA0)
4294790670 PID:400002 TID:10f0002 OSAXST1: >>> Loading Module 'k.ace.dll' (0x9937DDB0) at address 0xC12C0000-0xC1483000 in Process 'NK.EXE' (0x8239FAA0)
PB Debugger Loaded symbols for 'C:\WINCE600\OSDESIGNS\OSDESIGN1\OSDESIGN1\RELDIR\DEVICEEMULATOR_ARMV4I_DEBUG\K.ACE.DLL'
```

Check the log for other messages including the text UPDD, particularly when touching the screen. In the event of USB errors useful information is output in the log.

4) Enable UPDD debug log information

Additional UPDD debug information can also be seen by setting the tbupdd.ini debuglevel setting to 5.

```
TBUPDD.INI file
[UPDD]
Debuglevel=0x00000005
```

The USB component also shows more debug information when executed in a debug build.

Serial debug connection

If you wish to use a Serial connection to the Windows CE device for debug purposes we did document many years ago that there is a consideration that is not always made clear in the Microsoft documentation. A null modem cable is required, but this differs from a standard null modem cable in that the RI pin is connected straight through. Without this connection it will be impossible to make a serial connection from the NT host to the Win CE device. Nowadays most system integrators do not debug over serial so we are not sure if this is still a valid issue.

Port interface issues – important

Serial

Touch-screens may be connected to a CE device via a standard serial (COM). A CE image builder should bear in mind that the default CE image generated by platform builder might well make assumptions regarding the usage of such ports. E.g. debug output will be sent to the first physical COM port, preventing its use. By default, CE creates two com port instances, Com 0 and Com 1. Com 0 is used as the debug port and relates to the physical port com 1. Com 1 is therefore the first port that can be used by the touch screen that actually relates to the physical Com 2 port. Many customers have been unable to get their touch screens working with a default CE build until they have plugged the touch screen into com2 or changed the BIOS so that serial port is referenced as Com2, although the references in the CE build refer to com 1 !!!!

In some circumstances the CE builder will need to amend the CE configuration to alter the default serial port behavior. If you are not familiar with this procedure we have technical bulletin that covers this subject.

Port initialisation

We have discovered that with serial communications on some motherboards if we set up UART flow control options then we receive either no data or garbage. When this setup is omitted the serial port works as expected.   This is fine for serial controllers that do not need either a specific state or change of state in the CTS/RTS lines in order to operate.

We have introduced a device level setting called "IgnoreDCB" and if it is set to 0x00000001 then the com port flow control initialisation is skipped.

```
[updd\Parameters\1\]:IgnoreDCB = 0x00000001
```

We found that the TI OMAP Platform motherboard required this setting.
This is also used for the Zytronic ZXY100 controller on ARM hardware where setting DCB causes rubbish to be read

USB

The CE image must be amended to support the USB host controller (this is the system's USB host controller and NOT the USB touch controller). Consult the manufacturer's documentation and or Platform Builder help for details of how to achieve this with the particular model of hardware in use. A 3rd party driver might be required for the host controller, although this has not the case for the hardware we have tested so far. To ensure the CE system's USB host controller is functioning use a HID mouse prior to testing the USB touch controller.

PS/2

PS/2 support needs to be reinstated for the new driver as and when required….

Known issues

The list of known issues is as follows:

1) Could not ActivateDevice warning message in debug log with touch working fine

    When viewing the system debug log you may find an entry in a system where the touch is working fine that specifies:

    ```
    DEVICE!InitDeviceContent: can not Init Device content for 'Drivers\USB\LoadClients\2801_1\Default\Default\Tbupddceusb,error code 2404'
    !UPDD_USB: Could not ActivateDevice Drivers\USB\LoadClients\2810_1\Default\Default\Tbupddceusb 0x00000964
    ```

    This error is documented thus "System error code 2404 means "The device is in use by an active process and cannot be disconnected." This error code may also display as "ERROR_DEVICE_IN_USE" or as the value 0x964.

    We aren't quite sure why the "already in use" error occurs, but our suspicion is that it relates to the fact that the module load is attempted at two points.

    1) During the enumeration and matching phase (i.e. when the OS is looking for a matching driver).
    2) When theUSB system activates the driver.

    We think this message is simply a warning that the module is already loaded.
    We believe this only occurs in Win CE7.

    This error should not be ignored if the touch is not working!

2) After installing driver image worked once but failed to boot on subsequent reboots

    Customer determined that the image had increased in size and was too big for the flash disc allocated space. They increased the value associated with setting PLAT_IPSM_START_OFFSET and subsequent boots were OK.

3) Hot plug of device not working

    Hot plugging is not fully supported, in some cases this is due to limitations of the USBHC driver and / or USBHC hardware component. As of 23/02 improved hot plug supported added.

4) Missing \application data\updd folder and therefore no tbupdd.ini

    This caused issues with the driver. The user had used #includes in platform.reg, platform.bib and platform.dat. However it was not recognised in platform.dat, so tbupdd.dat was not included. As a consequence tbupdd.ini file was not included in the build.

Limitations

Known limitations to be addressed as required:

USB and serial (Oct 11 onwards) support only. Other interfaces to be added as required.
No sound support.
Dynamic rotate untested.
Hardware based eeprom storage not supported.
Limited testing of software based eeprom calibration storage.
No GUI settings dialog (UPDD Console) to allow for dynamic setting changes.

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com.